Modelling and solving the Sports Tournament Schediling (STS) problem

Hassen Said Ali (hassensaid.ali@studio.unibo.it)
Luwam Major Kefali (luwammajor.kefali@studio.unibo.it)
Hilina Fissha Woreta (hilina.woreta@studio.unibo.it)

1 Introduction

This report presents a few approaches to solving the Sports Tournament Scheduling (STS) problem. Using each approach we address both the optimization and the decision versions of the problem. And we were able to solve instances of scheduling up to 16 teams. The approaches implemented in this report are CP, SAT, SMT and MIP. Each team member handled the implementation of different approaches in the project. Hassen handled CP, Luwam handled both SAT and SMT while Hilina handled MIP approach

1.1 Problem Formalization

The common parameters across all approaches include:

- n: the number of teams, where n is even
- Teams $= \{1, \ldots, n\}$: the set of teams
- Weeks $= \{1, \ldots, n-1\}$: the set of weeks in the tournament
- Periods $= \{1, \ldots, n/2\}$: the set of time periods per week

The problem constraints require that each team plays every other team exactly once throughout the tournament, with each team playing exactly once per week. Additionally, each team can appear in a given period at most twice across all weeks.

In the optimization variant, we seek to balance the home and away games for fairness, minimizing the maximum deviation between the number of home games and away games for any team.

2 CP Model

2.1 Decision variables

The model employs two complementary views of the schedule:

Match view: The three-dimensional array match$[w, p, s]$ for $w \in$ Weeks, $p \in$ Periods, and $s \in \{1, 2\}$, where match$[w, p, 1]$ represents the home team and match$[w, p, 2]$ represents the away team playing in period p of week w. The domain of each element is Teams.

Opponent view: The two-dimensional array opponent$[t, w]$ for $t \in$ Teams and $w \in$ Weeks, where opponent$[t, w] = t'$ indicates that team t plays against team t' in week w. The domain is Teams.

Optimization variables: For the optimization variant, we introduce:

- home_games$[t]$ for $t \in$ Teams: a count variable with domain $[0, n-1]$ representing the number of home games played by team t
- max_dev: an integer variable representing the maximum deviation across all teams between twice their home games and their total games $(n-1)$

2.2 Objective function

For the decision variant, we seek any feasible schedule. For the optimization variant, the objective is to minimize max_dev, defined as:

$$\text{max\_dev} = \max_{t \in \text{Teams}} |2 \cdot \text{home\_games}[t] - (n-1)|$$

This objective minimizes the maximum imbalance between home and away games. The ideal value is 1 when n is even, as each team plays $n-1$ games total, making perfect balance impossible. The variable home_games[t] is computed using a global cardinality constraint on the home team positions in the match array.

2.3 Constraints

Channeling constraint: The match and opponent views are linked via:

$$\forall w \in \text{Weeks}, p \in \text{Periods} : \text{opponent}[\text{match}[w,p,1], w] = \text{match}[w,p,2] \land \text{opponent}[\text{match}[w,p,2], w] = \text{match}[w,p,1]$$

This ensures both views remain consistent throughout the search.

Self-play prevention: No team plays against itself:

$$\forall t \in \text{Teams}, w \in \text{Weeks} : \text{opponent}[t,w] \neq t$$

Round-robin constraint: Each team plays every other team exactly once, enforced using the `alldifferent` global constraint:

$$\forall t \in \text{Teams} : \texttt{alldifferent}([\text{opponent}[t,w] \mid w \in \text{Weeks}])$$

Weekly uniqueness: Each team plays at most once per week, enforced via:

$$\forall w \in \text{Weeks} : \texttt{alldifferent}([\text{match}[w,p,s] \mid p \in \text{Periods}, s \in \{1,2\}])$$

Period constraint: Each team appears in a given period at most twice across all weeks, enforced using the `global_cardinality_low_up` constraint:

$$\forall p \in \text{Periods} : \texttt{global\_cardinality\_low\_up}([\text{match}[w,p,s] \mid w \in \text{Weeks}, s \in \{1,2\}], \text{Teams}, [0,\dots,0], [2,\dots,2])$$

Home game counting: For optimization, we count home games using:

$$\texttt{global\_cardinality}([\text{match}[w,p,1] \mid w \in \text{Weeks}, p \in \text{Periods}], \text{Teams}, \text{home\_games})$$

Symmetry breaking constraints

The STS problem exhibits significant symmetry due to team renaming, week reordering, and period reordering. We impose the following symmetry breaking constraints:

First week fixing: We fix the schedule of the first week by assigning:

$$\forall p \in \text{Periods} : \text{match}[1,p,1] = 2p-1 \land \text{match}[1,p,2] = 2p$$

This eliminates symmetries arising from team relabeling.

First period fixing: In weeks where team 1 plays, we ensure team 1 is always the home team in the first period:

$$\forall w \in \text{Weeks} : (\text{match}[w,1,1] = 1 \lor \text{match}[w,1,2] = 1) \Rightarrow \text{match}[w,1,1] = 1$$

Ordering constraint: For the decision variant, we enforce that home teams have lower indices than away teams in all matches:

$$\forall w \in \text{Weeks}, p \in \text{Periods} : \text{match}[w,p,1] < \text{match}[w,p,2]$$

Note that this constraint is disabled in the optimization variant as it prevents achieving balanced home/away assignments.

These symmetry breaking constraints significantly reduce the search space without eliminating valid solutions.

2.4 Validation

Experimental design The CP model was implemented in MiniZinc and evaluated using three solvers: Chuffed, Gecode, and OR-Tools CP-SAT solver (labeled as "cp"). Experiments were conducted on instances with $n \in \{6, 8, 10, 12, 14, 16\}$ teams. A time limit of 300 seconds was imposed for each instance.

We evaluated four model configurations:

1. **Regular (reg):** Decision variant without search strategy
2. **Symmetry Breaking (sb):** Decision variant with all symmetry breaking constraints
3. **Optimization with regular (opt_reg):** Optimization variant without ordering constraint
4. **Optimization with symmetry breaking (opt_sb):** Optimization variant with first week and first period symmetry breaking only

Additionally, we tested a custom search strategy (labeled "ss") that:

- Uses sequential search, first assigning home teams, then away teams
- Employs `first_fail` variable selection with `indomain_split` or `indomain_random` value selection for home teams (with optional domain-weighted degree heuristic depending on the solver)
- Uses `first_fail` with `indomain_min` for away teams
- Applies geometric restart with factor 1.5 and base 500

All experiments were run on standard hardware with MiniZinc 2.9.4. Runtimes are reported in seconds, rounded to the nearest second.

Experimental results Tables 1–4 present the experimental results. For decision problems (Tables 1 and 3), cells show runtime in seconds. For optimization problems (Tables 2 and 4), cells show the best objective value found, with bold indicating proven optimality. "NA" indicates no solution was found within the time limit.

N	Chuffed	Chuffed+SB	CP-SAT	CP-SAT+SB	Gecode	Gecode+SB
6	0	0	0	0	1	0
8	0	0	0	0	NA	0
10	5	0	5	0	NA	0
12	23	1	22	1	NA	NA
14	NA	21	NA	21	NA	NA
16	NA	139	NA	139	NA	NA

Table 1: Decision variant runtime (seconds) without search strategy.

N	Chuffed	Chuffed+SB	CP-SAT	CP-SAT+SB	Gecode	Gecode+SB
6	**1**	**1**	**1**	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**	**1**	**1**
10	NA	**1**	NA	**1**	NA	**1**
12	NA	**1**	NA	**1**	NA	NA
14	NA	NA	NA	NA	NA	NA
16	NA	NA	NA	NA	NA	NA

Table 2: Optimization variant: best objective value found (bold = optimal).

N	Chuffed+SS	Chuffed+SB+SS	CP-SAT+SS	CP-SAT+SB+SS	Gecode+SS	Gecode+SB+SS
6	0	0	0	0	0	0
8	NA	0	NA	0	2	0
10	0	0	0	0	NA	0
12	NA	1	NA	1	NA	1
14	NA	2	NA	2	NA	5
16	NA	66	NA	66	NA	NA

Table 3: Decision variant runtime (seconds) with custom search strategy.

N	Chuffed+SS	Chuffed+SB+SS	CP-SAT+SS	CP-SAT+SB+SS	Gecode+SS	Gecode+SB+SS
6	**1**	**1**	**1**	**1**	**1**	**1**
8	NA	**1**	NA	**1**	**1**	**1**
10	3	**1**	3	**1**	NA	NA
12	NA	NA	NA	NA	NA	NA
14	NA	NA	NA	NA	NA	NA
16	NA	NA	NA	NA	NA	NA

Table 4: Optimization variant with search strategy: best objective value (bold = optimal).

Analysis The experimental results demonstrate several key findings:

Symmetry breaking effectiveness: shows improvement accross the board for any combination of solvers. In the decision variant without search strategy (Table 1), symmetry breaking reduces runtime from 5 to 0 seconds for $n = 10$, and from 23 to 1 second for $n = 12$ in Chuffed and CP-SAT.

Optimization challenges: Without symmetry breaking, only instances up to $n = 8$ can be solved optimally (Table 2). With symmetry breaking, we can solve up to $n = 12$ optimally, achieving the theoretical minimum deviation of 1. No solver could solve $n \geq 14$ within the time limit in optimization mode.

Search strategy impact: The custom search strategy had a mixed bag results, for the decision making models when combined with symmetry breaking it slashed the times of solutions specially for the chuffed model for 16 instances

Solver comparison: Chuffed and CP-SAT demonstrate nearly identical performance across all configurations, suggesting similar underlying solving strategies. Gecode shows weaker performance, frequently timing out on larger instances even with symmetry breaking. This is particularly evident in Tables 1 and 2, where Gecode fails on $n \geq 8$ without symmetry breaking and $n \geq 12$ with symmetry breaking in the optimization variant.

Scalability: The problem exhibits exponential growth in difficulty. The largest instance solved to optimality is $n = 12$. Instance $n = 16$ represents the scalability limit for the decision variant with symmetry breaking, requiring 139 seconds.

Overall, the combination of symmetry breaking constraints and appropriate solver selection (Chuffed or CP-SAT) provides the most robust approach to the STS problem, enabling optimal solutions up to moderate instance sizes.

3 SAT Model

This section presents the propositional encoding of the STS problem. The encoding is implemented in Python using the Z3 API for direct SAT solving, and translated to DIMACS CNF to be solved also by the external solver Glucose. All models share the common STS definitions of teams, weeks, and periods given in Section 1. We consider both the pure decision version and the optional fairness optimisation.

3.1 Decision variables

Let $T = \{1, \dots, n\}$ be the set of teams, $W = \{0, \dots, n-2\}$ the set of weeks, and $P = \{0, \dots, \frac{n}{2} - 1\}$ the set of periods per week, with n even. For each unordered pair of distinct teams $\{i, j\}$ with $i < j$, period $p \in P$ and week $w \in W$, we introduce the Boolean variable

$$M_{i,j,p,w} = \begin{cases} \texttt{True} & \text{iff teams } i \text{ and } j \text{ play in period } p \text{ of week } w, \\ \texttt{False} & \text{otherwise.} \end{cases}$$

These literals encode the placement of matches and form the core of the SAT model, both in the direct Z3 encoding and in the DIMACS translation.

For the Z3 optimisation model, we additionally introduce home–away variables for each unordered pair $\{i, j\}$ with $i < j$:

$$H_{i,j,p,w} = \begin{cases} \texttt{True} & \text{iff } i \text{ is the home team and } j \text{ is the away team in } (p, w), \\ \texttt{False} & \text{otherwise.} \end{cases}$$

In the DIMACS encoding these variables are declared but only the structural constraints over $M_{i,j,p,w}$ are used, since Glucose is employed purely for feasibility checking.

For the fairness constraint, the Z3 model introduces auxiliary literals

$$\mathsf{HOME}_{t,j,p,w}$$

meaning that team t plays at home against j in slot (p, w). Each such literal is equivalently defined in terms of M and H depending on whether t is the first or second team of the unordered pair.

3.2 Objective function

The optional objective is to minimise the maximum imbalance between home and away games over all teams. Each team plays exactly $n - 1$ games, so it cannot be perfectly balanced; the theoretical imbalance satisfies

$$1 \le \left|\text{HomeGames}_t - \text{AwayGames}_t\right| \le n - 1, \quad \forall t \in T,$$

as in the project description. :contentReference[oaicite:0]index=0

In our SAT encoding, optimisation is implemented by bounding the number of home games per team. For a fixed integer parameter $\Delta \ge 0$, we define for each team t:

$$\text{HomeGames}_t = \sum_{j \ne t} \sum_{p \in P} \sum_{w \in W} \mathsf{HOME}_{t,j,p,w}.$$

Since $\text{HomeGames}_t + \text{AwayGames}_t = n - 1$ for every t, the imbalance constraint

$$\left|\text{HomeGames}_t - \text{AwayGames}_t\right| \le \Delta$$

is equivalent to the pure cardinality bounds

$$\text{minHome} = \frac{(n-1) - \Delta}{2}, \qquad \text{maxHome} = \frac{(n-1) + \Delta}{2},$$

$$\text{minHome} \le \text{HomeGames}_t \le \text{maxHome}, \quad \forall t \in T.$$

These bounds are encoded using Z3 cardinality constraints `AtLeast` and `AtMost` on the literals $\mathsf{HOME}_{t,j,p,w}$. The optimal value $\Delta^\star$ is found by a binary search procedure `binary_search_max_diff`, which repeatedly calls `build_model` with different values of Δ and checks satisfiability. In this way, SAT is used as a feasibility oracle for successively tighter fairness bounds.

Glucose is never used for optimisation: it only solves the pure decision model (with or without symmetry breaking) on the DIMACS CNF.

3.3 Constraints

All structural constraints are expressed directly on the match placement variables $M_{i,j,p,w}$ and correspond exactly to those implemented in `build_model` and mirrored in the DIMACS generator `sat_dimacs.py`.

C1. Each pair of teams plays exactly once. For every unordered pair $\{i, j\}$ with $i < j$:

$$\sum_{p\in P}\sum_{w\in W} M_{i,j,p,w} = 1.$$

C2. Each team plays exactly once per week. For every team $t \in T$ and week $w \in W$:

$$\sum_{p\in P}\sum_{j\in T\setminus\{t\}} M_{\min(t,j),\max(t,j),p,w} = 1.$$

C3. Exactly one match per period and week. For each period $p \in P$ and week $w \in W$:

$$\sum_{i<j} M_{i,j,p,w} = 1.$$

C4. Period frequency constraint. For every $t \in T$ and period $p \in P$:

$$\sum_{j\in T\setminus\{t\}}\sum_{w\in W} M_{\min(t,j),\max(t,j),p,w} \;\le\; 2.$$

Symmetry breaking

The STS problem exhibits strong symmetries due to team renaming and joint home/away flipping. When symmetry breaking is enabled (`use_symmetry = True`), we impose the following constraints:

SB1. Canonical first week. Week 0 is fixed by imposing

$$M_{2p+1,2p+2,p,0} = \texttt{True}, \qquad H_{2p+1,2p+2,p,0} = \texttt{True},$$

for all $p \in P$ with $2p + 2 \le n$.

SB2. Fixed opponent order for team 1. For every $w \in W$ with $w + 2 \le n$:

$$\sum_{p\in P} M_{1,w+2,p,w} = 1.$$

SB3. Team 1 always at home.

$$M_{1,j,p,w} \;\Rightarrow\; H_{1,j,p,w}, \quad \forall j \in T \setminus \{1\},\; p \in P,\; w \in W.$$

Optimisation (fairness) constraints

When a bound Δ is given, for each team t we introduce the literals $\mathsf{HOME}_{t,j,p,w}$ defined by

$$\mathsf{HOME}_{t,j,p,w} \;\Leftrightarrow\; \begin{cases} M_{t,j,p,w} \wedge H_{t,j,p,w}, & \text{if } t < j, \\ M_{j,t,p,w} \wedge \neg H_{j,t,p,w}, & \text{if } t > j. \end{cases}$$

and enforce

$$\text{minHome} \le \sum_{j\ne t,p,w} \mathsf{HOME}_{t,j,p,w} \le \text{maxHome}.$$

3.4 Validation

3.5 Experimental design

The SAT models were implemented in Python using Z3 for direct solving and a separate script to generate DIMACS CNF files for Glucose. All experiments were run inside the project Docker container on a laptop with a recent Intel CPU and 16 GB RAM. A time limit of 300 seconds per run was enforced by setting the Z3 timeout and terminating external Glucose calls, in accordance with the project specifications. :contentReference[oaicite:1]index=1

We considered instances with

$$n \in \{6, 8, 10, 12, 14, 16\},$$

and evaluated four approaches: Z3, Z3+SB, Glucose, and Glucose+SB. For each instance we solved both the decision version and, using only Z3, the optimisation version via binary search on Δ.

3.6 Experimental results

n	**Z3**	**Z3(SB)**	**Z3(OPT)**	**Z3(OPT+SB)**	**Glucose**	**Glucose(SB)**
6	0	0	N/A	0	0	0
8	0	0	N/A	2	2	2
10	0	0	N/A	15	38	34
12	19	11	N/A	N/A	N/A	N/A
14	144	N/A	N/A	N/A	N/A	N/A
16	N/A	N/A	N/A	N/A	N/A	N/A

Table 5: Execution times (in seconds) for SAT and OPT models, with and without symmetry breaking, for values of n from 6 to 16. "N/A" indicates that no solution was found within the 300-second time limit.

4 MIP Model

This section presents the Mixed-Integer Linear Programming (MIP) formulation used for the Sports Tournament Scheduling (STS) problem. All models share the global definitions introduced in Section 1 (teams, weeks, periods, and schedule structure). Four variants of the MIP formulation were developed:

- Plain model,
- Symmetry-breaking model,
- Implied-constraints model,
- Optimization model (fairness objective).

All models are linear and solver-independent.

4.1 Decision variables

All model variants use the binary variable

$$y_{ijwp} = 1 \iff \text{team } i \text{ plays at home against team } j \text{ in week } w \text{ period } p,$$

for all $i \neq j$, $w \in W$, and $p \in P$.

The optimization model introduces additional linear auxiliary variables:

$$h_t,\ a_t,\ d_t \in \mathbb{Z}_{\geq 0} \quad \forall t, \qquad F \in \mathbb{Z}_{\geq 0},$$

representing home games, away games, imbalance, and maximum imbalance.

4.2 Objective function

Decision models. The plain, symmetry-breaking and implied models solve the feasibility (decision) version:

$$\min 0.$$

Fairness optimization model. The goal is to minimize the maximum home/away imbalance:

$$\min F.$$

This is linearised using:

$$h_t = \sum_{j \neq t,\, w,p} y_{tjwp}, \qquad a_t = \sum_{j \neq t,\, w,p} y_{jtwp},$$

$$d_t \geq h_t - a_t, \qquad d_t \geq a_t - h_t, \qquad F \geq d_t.$$

4.3 Constraints

All four variants include the following linear STS constraints:

No self-matches.

$$y_{iiwp} = 0 \quad \forall i, w, p.$$

Exactly one match per period/week slot.

$$\sum_{i \neq j} y_{ijwp} = 1 \quad \forall w, p.$$

Each pair meets exactly once.

$$\sum_{w,p} (y_{ijwp} + y_{jiwp}) = 1 \quad \forall i < j.$$

Each team plays exactly once per week.

$$\sum_{j \neq t, p} (y_{tjwp} + y_{jtwp}) = 1 \quad \forall t, w.$$

Period frequency constraint. Each team appears at most twice in the same period:

$$\sum_{j \neq t, w} (y_{tjwp} + y_{jtwp}) \leq 2 \quad \forall t, p.$$

Symmetry-breaking constraints

Week 1 is fixed to the canonical pattern:

$$y_{(2p-1)(2p)1p} = 1 \quad \forall p.$$

Implied constraints

These strengthen the model without removing feasible schedules:

$$\sum_{j \neq t, w, p} (y_{tjwp} + y_{jtwp}) = n - 1 \quad \forall t,$$

$$\sum_{i \neq j, w} y_{ijwp} = n - 1 \quad \forall p.$$

4.4 Validation

All models were implemented in AMPL and evaluated using the MIP solvers CPLEX and Gurobi with a time limit of 300 seconds. We tested instances with $n \in \{6, 8, 10, 12, 14\}$. For each instance, we recorded solver time, termination status, objective value (for the fairness model), and the extracted schedule when available.

Decision model results. Table 6 shows solving times for the feasibility models, where `NA` indicates a timeout.

Table 6: Decision models

N	MIP_implied_cplex	MIP_implied_gurobi	MIP_plain_cplex	MIP_plain_gurobi	MIP_symmetry_cplex	MIP_symm
6	0	0	0	0	0	
8	0	0	0	0	0	
10	0	1	0	0	0	
12	8	11	9	12	12	
14	NA	27	55	23	278	1

Table 7: Decision models

N	MIP_implied_cplex	MIP_implied_gurobi	MIP_plain_cplex	MIP_plain_gurobi	MIP_symmetry_cplex	MIP_symm
6	0	0	0	0	0	
8	0	0	0	0	0	
10	0	1	0	0	0	
12	8	11	9	12	12	
14	300	27	55	23	278	1

Decision model raw times.

Optimization model results. Table 8 reports the optimal fairness value F (null $\rightarrow$ NA).

Table 8: Optimization model

N	MIP_opt_cplex	MIP_opt_gurobi
6	1.0	1.0
8	1.0	1.0
10	1.0	1.0
12	1.0	1.0
14	1.0	NA

Optimization model raw values.

Discussion. For small instances ($n \leq 10$), all variants solve instantly. For $n = 12$, both solvers require more time, with CPLEX generally faster. At $n = 14$, only Gurobi solves all decision variants; CPLEX times out on the implied model. The optimization model consistently finds the optimal imbalance $F = 1$ except when Gurobi times out at $n = 14$.

These results confirm that the MIP model is correct, scalable for moderate n, and that symmetry-breaking significantly improves solving time.

Table 9: Optimization model

N	MIP_opt_cplex	MIP_opt_gurobi
6	1.0	1.0
8	1.0	1.0
10	1.0	1.0
12	1.0	1.0
14	1.0	None

5 Conclusions

The experimental study shows clear differences in how CP, SAT, and MIP handle the Sports Tournament Scheduling problem as instance sizes grow. CP is the strongest overall for feasibility: with symmetry breaking and a suitable search strategy, both Chuffed and CP-SAT solve all decision instances up to = 16 n=16, and remain competitive on the optimisation variant up to = 12 n=12. SAT performs well on small and medium instances, with symmetry breaking dramatically reducing solving time, but its optimisation version becomes difficult beyond = 10 n=10. External SAT solving with Glucose is useful for feasibility but does not scale to larger instances.

MIP provides a clean and expressive formulation and solves all instances up to = 12 n=12 efficiently, with Gurobi extending feasibility to = 14 n=14. The fairness optimisation consistently returns the theoretical optimum = 1 F=1 whenever a solution is found.

Overall, the results highlight the expected scalability limits of each paradigm: CP offers the best raw performance on decision problems, SAT benefits heavily from symmetry breaking but struggles on optimisation, and MIP handles moderate instance sizes with strong performance, especially under symmetry-aware formulations. Together, the three approaches validate the correctness of the modelling and provide complementary strengths across the tested instances.

Authenticity and Author Contribution Statement

The work is our own and we did not copy it from someone else. Ideas that we took from literature search and existing implementations are cited properly.

Author's contribution to the work:
Hassen Said Ali on CP modelling.
Luwam Major Kefali on SAT modeling.
Hilina Fissha Woreta on MIP.

References

[1] J. van Doornmalen, C. Hojny, R. Lambers, and F. C. R. Spieksma. *Integer programming models for round robin tournaments.* Eindhoven University of Technology, 2021.

[2] D. Goossens and J. Beliën. *Teaching Integer Programming by Scheduling the Belgian Soccer League.* INFORMS Transactions on Education, 2022.

[3] I. Hucijak and T. Kačer. A SAT-based approach for round-robin tournament scheduling. *Operations Research Perspectives*, Elsevier, 2019.

[4] C. Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *CP 2005.*

[5] J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-Breaking Predicates for Search Problems. In *KR 1996.*

[6] J.-F. Puget. Symmetry Breaking Revisited. In *CP 2005.*

[7] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, 2009.

[8] L. M. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS 2008*.

[9] A. Reynolds et al. cvc5: A Next-Generation SMT Solver. In *CAV 2020*.

[10] G. Gange, P. J. Stuckey, and P. Van Hentenryck. Solving Scheduling Problems via SMT. In *CPAIOR 2011*.